FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Gestão Documental

Miguel José Silva Gomes

VERSÃO DE TRABALHO

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Orientador: Ana Cristina Ramada Paiva

27 de Junho de 2017

Resumo

O presente projeto descreve o desenvolvimento de um sistema de gestão documental realizado no decorrer do estágio na empresa Finantech - Sistemas de Informação, S.A. Trata-se de uma prova de conceito com a finalidade de estandardizar o arquivamento de documentos e mitigar as perdas relativas ao tempo de procura e perda de documentação.

Ao longo da ultimas décadas, decorreu uma transição drástica ao nível da documentação ao nível empresarial. A documentação deixou quase de existir fisicamente e passou a existir digitalmente, bem como os métodos de organização que as empresas adotavam na altura ficou obsoleto. Ainda hoje em dia é perfeitamente normal não existir sistemas de gestão documental no seio empresarial devido aos recursos necessários para comprar ou desenvolver um.

No caso da Finantech, não existe uma forma standard de organização e classificação documental para toda a empresa. Existe dentro de cada departamento boas normas de como arquivar os documentos nos diretórios por eles definidos. Esta metodologia causa duplicação de documentação pela empresa, uma vez que cada departamento tem o seu repositório e guarda com as suas próprias regras. Pode até levar a perda de documentação quando endividamento arquivado.

À luz disto, a Finantech pretende realizar uma prova de conceito que envolve tanto o estudo e modelação de uma ontologia capaz de gerir a documentação da empresa, bem como a construção de uma aplicação web que proporcione a organização que a empresa precisa.

Deste modo, adotou-se uma metodologia de protótipo dada a elevada dependência de interações para analisar as bases de dados, catalogar os diferentes tipos de documentos por departamento, refinar a estrutura da ontologia e definir o design da aplicação web com colaboradores da Finantech.

A ontologia é a base do projeto, onde se definiu o modelo estrutural do projeto. A classificação dos documentos segundo vários metadados relevantes à catalogação dos diferentes documentos na Finantech, era a chave do projeto. Para isso, foi fulcral o mapeamento da informação essencial para a documentação, como os clientes e os projetos, da base de dados para a ontologia desenvolvida. Por fim, mapear a ontologia em um servidor capaz de processá-la.

É preponderante, que a comunicação entre base de dados e servidor triplo seja em tempo real. Assim, caso seja adicionado um novo cliente, a aplicação seja capaz de arquivar a documentação relativa a esse cliente.

A aplicação web é composto por duas camadas ambas desenvolvidas em ASP.NET. O *web service*, onde são levantas as interrogações à ontologia. O *Client* que realiza o tratamento dos dados com o intuito de permitir a visualização das respostas geradas pelas interrogações.

O trabalho desenvolvido foi avaliado junto a colaboradores de vários departamentos da Finantech, de forma a verificar se correspondia a uma melhoria das práticas correntes.

Abstract

This project describes the development of a management system document carried out during the internship at Finantech - Sistemas de Informação, SA. This is a proof of concept in which the purpose is standardizing the archiving of documents and by doing that mitigate the time losses related to research and also loss of documentation.

Over the last few decades, there has been a drastic transition at business-level documentation. The documentation almost ceased physically and came into digitally existence. As well, the organizational methods that companies adopted at the time became obsolete. Even nowadays it is perfectly normal not to have document management systems in the business sector due to amount of investment needed to buy or to develop one.

In the Finantech case, doesn't exist a standard form of organization and documentary classification for all the company. However, within each department, there are good rules on how to file the documents in the directories defined by them. This methodology causes duplication of documentation by the company, since each department has its repository and keeps with its own rules. It may even lead to loss of documentation when filing indebtedness. Thereby, Finantech intends to carry out a proof of concept that involves both the study and modeling of an ontology capable of managing the company's documentation, as well as the construction of a web application that provides the organization that the company needs.

Thus, a prototype methodology was adopted because of the high reliance on interactions to analyze the databases, catalog the document types by department, refine the structure of the ontology and define the web application design with Finantech's collaborators.

The ontology is the basis of the project, where the structural model of the project was defined. The classification of the documents, according to various metadata relevant to the cataloging of the different documents in Finantech, was the key to the project. For this, it was essential to map the essential information for the documentation (as clients and projects) of the database for the developed ontology. The final stage is to map the ontology into a server that can process it.

It is preponderant, that the communication between database and triple server is in real time, for example, if a client is added, the application should be able to archive the documentation as relative to that new client. The web application consists of two layers both developed in ASP.NET. The web service, where the queries are raised to the ontology and the Client, performs the data treatment in order to allow the visualization of the answers generated by the interrogations.

The work developed was evaluated with the employees of all of Finantech's departments, in order to verify if it corresponded to an improvement of the current practices.

Agradecimentos

A elaboração deste projeto só foi possível graças ao apoio que me foi dado. Neste sentido, cabe-me agradecer à Professora Doutora Ana Paiva pela sua disponibilidade no esclarecimento de qualquer dúvida, pelas suas sugestões e aconselhamento. À empresa Finantech e seus colaboradores agradeço toda a atenção, compreensão e acolhimento que senti na mesma, de facto, sem a Finantech este projeto não seria desenvolvido. Agradeço também a todos aqueles que de forma mais ou menos direta tiveram impacto no resultado final deste projeto.

A todos, obrigado!

Miguel Gomes

"A persistência é o menor caminho do êxito."

Charles Chaplin

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

IRI	Internationalized Resource Identifier
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SKOS	SKOS Simple Knowledge Organization System
SQL	Structured Query Language
XML	eXtensible Markup Language
W3C	World Wide Web Consortium
ODBC	Open Database Connectivity
JSON	JavaScript Object Notation
API	Application Programming Interface
MVC	Model-view-controller

Capítulo 1

Introdução

A Finantech é uma empresa de sistemas de informação, principalmente focada no desenvolvimento de aplicações bem como serviços na área dos mercados financeiros. Conta com mais de 20 anos de experiência no sector e fortes parceiros a nível mundial, exemplos da Microsoft e da Oracle. A Finantech tem como principais clientes bancos de investimento, bancos comercias e *brokers*, com especial destaque ao mercado nacional.

Em termos de gestão documental, a Finantech, só possui regras dentro de equipas e departamentos. Consequentemente, verifica-se duplicação de documentos no próprio sistema, perda de documentação, dificuldade em encontrar documentos e desalinhamento ou falta de versões. As normas de gestão baseiam-se na organização de documentos por diretórios distribuídos por áreas e cada departamento adota a sua política arquivista.

A Finantech pretende com esta dissertação realizar um caso de estudo de tecnologias disponíveis e acessíveis para que seja possível a implementação de um sistema de gestão documental. O sistema passaria por desenvolver uma aplicação web capaz de organizar e classificar os documentos pelo que estes representam e não pelo diretório onde se encontram inseridos. No final comparar os resultados alcançados com o estudo realizado com a metodologia existente.

1.1 Objetivos

O principal objetivo da dissertação recai no desenvolvimento de uma aplicação de gestão documental, tendo em conta a organização, a visão e a base de dados estrutural apresentada pela Finantech.

Deste modo identificou-se vários objetivos imprescindíveis para atingir o objetivo final:

1. Análise e conceção de um modelo documental face a Finantech;
2. Construção de um ontologia no âmbito da documentação na Finantech;
3. Desenvolvimento de uma aplicação web capaz de estandardizar a documentação modelada na ontologia;
4. Validação do sistema desenvolvido junto dos colaboradores da Finantech.

1.2 Estrutura da dissertação

A estruturação deste documento segue uma organização hierárquica divida por capítulos, subcapítulos e sub-subcapítulos, sendo composta na totalidade por 6 capítulos.

No capitulo introdutório, apresenta-se a empresa na qual decorreu os estágio e o ponto de situação da mesma face ao âmbito da dissertação. Determina-se os objetivos essências do projeto e descreve-se a estrutura do documento.

O corpo do documento é constituído por 4 capítulos, o primeiro contém estado da arte. O qual relata a pesquisa, no seio científico, em termos de Gestão Documental e Ontologia.

Os seguintes capítulos do corpo, apresentam um cariz mais prático. No primeiro descreve os procedimentos para a modelação do sistema, dando grande foco a construção da ontologia. O segundo recai sobre o desenvolvimento da aplicação, que se encontra dividido em dois subcapítulos o *web service* e o *front-end*. Por fim, o ultimo capítulo apresenta os argumentos correspondestes a validação da aplicação desenvolvida.

Capítulo 2

Enquadramento

Este capítulo representa o estudo realizado, a priori, a partir do conhecimento recolhido no seio científico, de forma a consolidar os conceitos importantes para o projeto. O enquadramento está dividido em dois grandes temas a Gestão Documental e Ontologia.

2.1 Gestão Documental

Os documentos desempenham um papel fulcral no seio empresarial em vários níveis. Por esse motivo é essencial analisar e organizar os documentos perante a informação, o seu âmbito e a estrutura empresarial envolvente [3]. Por isso, o documento foi objeto de estudo nas Ciências Documentais (abordagem arquivística), com o intuito de desenvolver métodos de caracterização e classificação. Por outro lado, a Informática concentrou-se no estudo da informação inerente e não à sua origem [4].

A Gestão Documental, considerando uma abordagem arquivística, tem como requisitos criar, organizar, utilizar, conservar, avaliar, selecionar e eliminar documentos. O estudo destas metodologias têm vindo a evoluir desde a revolução francesa até os dias de hoje. O que demonstra a grande necessidade que as organizações apresentam nos processos de produção e armazenamento diários [5].

2.1.1 Mercado

Perante a volatilidade económica da ultima década, a eficiência e otimização dos processos tornaram-se aspetos fulcrais da indústria. Devido a estes fatores, uma das áreas que mais evoluiu, no âmbito empresarial, foi a gestão dos conteúdos. Esta tem como finalidade fomentar a eficiência das tarefas dos trabalhadores e a partilha e criação de informação [1].

De acordo com um estudo realizado pela AIIM em 2011 "Our tracking over the last few years shows that lack of confidence in the accuracy, accessibility and trustworthiness of electronic information has remained at 40%". No mesmo estudo, perguntaram às empresas qual seria o principal motivo para implementar um serviço de gestão de conteúdos, o seguinte diagrama expõe as respostas. [1].

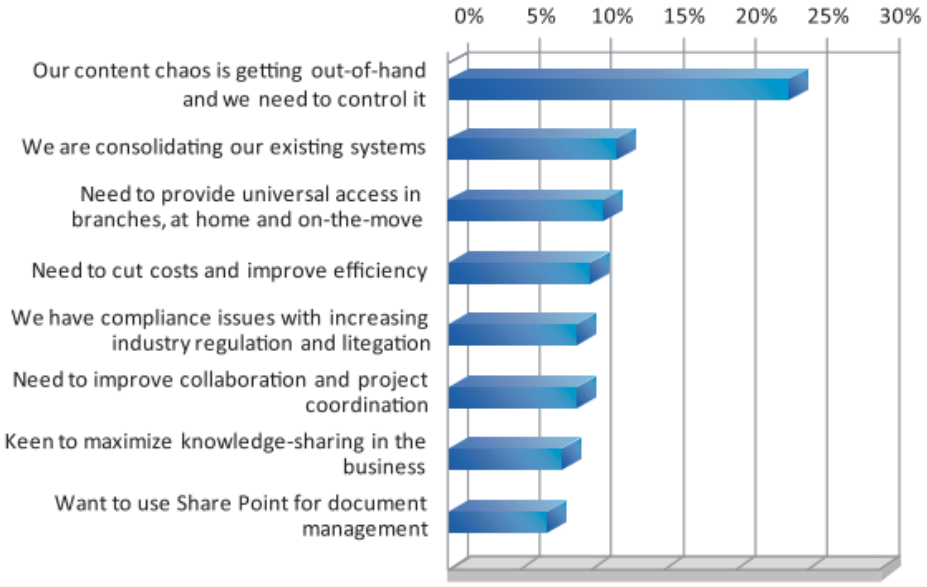

Figura 2.1: Motivo para comprar o serviço [1]

Como se pode verificar, o maior problema que as empresas em 2011 apresentavam era a capacidade de organizar os conteúdos.

2.1.2 Metadados

A acumulação e desorganização de documentos é uma lacuna comum a nível empresarial. É fundamental existir informação e partilhá-la pela organização, para isso é importante existir os meios de consulta, caso contrário, torna-se num recurso sem valor, ou seja, desperdício [3]. Posto isto, faz todo o sentido identificar todo o tipo de documentos, para que seja possível reutilizá-lo ou consultá-lo mais tarde.

Como se identificam todos os documentos? A abordagem tradicional recorre a metadados. Metadados não são mais nem menos do que dados sobre dados (para contextualizar com o tema dados sobre os documentos) que são armazenados e geridos numa base de dados ou repositório. Têm propósito de armazenar, organizar, gerir e partilhar [6].

Um fator a considerar é a classificação que designam um determinado conceito, usualmente associado a uma estrutura como *thesaurus*, taxonomia e ontologia. Estas estruturas possuem diferentes tipos de relações e axiomas e são desenhadas com o objetivo de conceber uma aplicação [6].

2.1.2.1 Dublin Core

Dublin Core é um projeto baseado em metadados, cujo objeto de conceção é o documento [7]. Desenhado sob a forma de ontologia, o Dublin Core é capaz de classificar e catalogar *Document-like objects* (textos, mapas, imagens, etc) [7], foi produto de um consenso interdisciplinar e inter-

nacional. A estrutura desenvolvida em RDFs, é composta por 15 elementos descritivos, os quais estão listados na seguinte tabela: [8].

DCMES Element	Element Refinements(s)	Element Encoding Schemes(s)
Date	Alternative	-
Creator	-	-
Subject	-	LCSH MeSH DDC LCC UDC
Description	Table of Contents Abstract	-
Publisher	-	-
Contributor	-	-
Date	Created Valid Available Issued Modified	DCMI Period W3C-DTF
Type	-	DCMI Type Vocabulary
Format	Extent	-
	Medium	IMT
Identifier	-	URI
Source	-	URI
Language	-	ISO 639-2 RFC 1766
Relation	Is Version Of Has Version Is Replaced By Replaces Is Required By Requires Is Part Of Has Part Is Referenced By References Is Format Of Has Format	URI
Coverage	Spatial	DCMI Point ISO 3166 DCMI Box TGN
	Temporal	DCMI Period W3C-DTF
Rights	-	-

Figura 2.2: Elementos classificadores do Dublin Core [2]

2.2 Ontologia

Nos dias de hoje, devido ao aumento exponencial dos dados disponíveis, existe a necessidade de criar técnicas de organização de dados, melhorando o tratamento de dados em termos de seleção, processamento, recuperação e disseminação. Para combater este problema foram desenvolvidas, ao longo dos anos, vários tipos de estruturas, que podem ser organizadas por termos (dicionários), classificações (taxonomias) e conceitos (ontologias).

A palavra ontologia é originária do grego concebida por Aristóteles, na antiguidade era conhecida como a ciência ou o estudo de ser, mais especificamente *"onto"* significa "ser" e *"logos"* significa "ciência". Nesta primeira conceção a palavra "ser" possuía dois significados, complementares, algo que é, ou seja, uma entidade, ou o que define ser ou existir? A junção destas duas vertentes designa-se ontologia filosófica. Por sua vez, o significado filosófico diverge da definição aceite dentro da comunidade científica na área da ciência de informação [9].

2.2.1 Definição Científica

Ontologia é uma estrutura de representação de conhecimento com o intuito de partilhar informação para uma determinada comunidade. Contudo, o que representa "informação" neste contexto? Segundo Stair e Reynolds (2001), "a informação pode ser considerada dados tornados mais úteis através da aplicação do conhecimento" - [10]. Considerando esta afirmação, Beynon-Davies (2002) caracteriza a informação como dados interpretados sobre um contexto e o conhecimento deriva da informação e das relações entre elas bem como o conhecimento existente [11]. Na comunidade científica, o estudo relativo a ontologia está divida em duas áreas de pesquisa: inteligência artificial (partilha de conhecimento entre sistemas) e base de dados (modelos de dados desenvolvido para a industria) [12].

Na literatura são apresentadas diversas definições de ontologias e existem contradições. A definição mais generalista define uma ontologia como um modelo de dados que representa um conjunto de conceitos de um domínio. Ontologia é constituída por um conjunto de classes com vários níveis hierárquicos, cada uma apresenta um conceito sobre um determinado domínio, e conforme o nível hierárquico é um conceito mais vago (quanto mais alto for o nível) ou mais específico (quanto mais baixo). Para além de classes existem as relações e regras que permitem a interpretação do conhecimento [13] [14].

A aplicação desta estrutura tem como finalidade a partilha de informação, entre os agentes de comunicação, nós, seres humanos, ou sistemas computacionais capazes de interpretar a inferência ontológica. Outro aspecto importante, é a capacidade reusabilidade e organização de conhecimento [15]. Todos estes aspectos têm de estar obrigatoriamente presentas para denominar-se ontologia.

Por outro lado, se a partilha da conceptualização não for um fator a considerar e apenas utilizar a estrutura conceção, por exemplo, caso se trate de uma aplicação especifica para uma empresa com objetivos lucrativos, a comunidade cientifica dá o nome de modelos de dados em vez de ontologia [16].

Engenharia de ontologias estuda os métodos e metodologias, capazes de distinguir um domínio com alto nível de abstração e capturar, o mais claramente possível os termos, com a finalidade de construir uma ontologia. Esta trata de analisar a linguagem e ferramenta que se adequa. Um desenvolvimento ambicioso que advém da implementação de ontologias é a *Semantic Web*.

2.2.2 Linguagens

Para uma ontologia ser compreendida tem de estar representada de alguma forma. Uma ontologia pode ser concebida em linguagens de alto nível como UML ou até linguagens naturais, contudo só o ser humano consegue processar tal informação. Dentro da variedade de linguagens existem *Logical Languages, Frame based Languages* e *Graph based Languages* [17]. Das quais se destaca OWL e SPARQL que se podem basear em vocabulários RDF/XML [18], RDF, RDFS [19], TURTLE ou JSON-LD [20]. De salientar a possibilidade de traduzir os vocabulários uns para os outros sem danificar o conteúdo dos grafos.

2.2.2.1 RDF Resource Description Framework

RDF é modelo de representação de dados estandardizado na *Web* desenvolvida pela W3C. É uma estrutura com o propósito de facilitar interligação de dados, o que torna ideal para desenvolver tecnologias como OWL ou SKOS. A sintaxe deste modelo é composto por sujeito (IRI ou nós vazios), predicado (IRI) e objeto (IRI, *literal* ou nós vazios) também conhecida por triplos, podem ser representado em grafos [21].

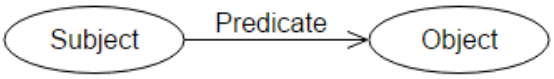

Figura 2.3: Representação de um triplo em grafo

IRI (exemplo: `http://dbpedia.org/resource/7_(number)`) ou *literal* (exemplo: "7"ou "7"8sd:integer) são chamados de recursos, que podem ser algo físico, um documento, um conceito abstrato, um número ou uma string. Os nós vazios são exatamente a mesma coisa só que são identificadores locais.

2.2.2.2 SPARQL Query Language

SPARQL é a linguagem que permite manipular conteúdos de grafos RDF na internet ou num armazenamento RDF, ou seja, são queries de triplos. Esta linguagem suporta XML, JSON, CSV, TSV, RDF Schema ou OWL. No próximo exemplo está representada uma *query* à ontologia *foaf (friend of a friend)*, onde se pretende verificar se a Alice tem um amigo com o nome de Snoopy [22].

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
WHERE {
    <http://example.org/alice#me> foaf:knows [ foaf:name ?name ] .
    SERVICE <http://dbpedia.org/sparql>
    { <http://dbpedia.org/resource/Snoopy> foaf:name ?name }
}
```

2.2.2.3 OWL Web Ontology Language

OWL, como o próprio nome indica, é uma linguagem para ontologias, criada para facilitar a construção e partilha de ontologia pela a *web*, com objetivo de tornar a informação web acessível às maquinas. Foi desenvolvida sobre uma estrutura RDF, por isso a informação criada nesta linguagem ser exportada como documentos RDF. OWL é composto no mínimo por uma sintaxe e uma semântica, por isso a próxima imagem poderia ser simplificada. [23].

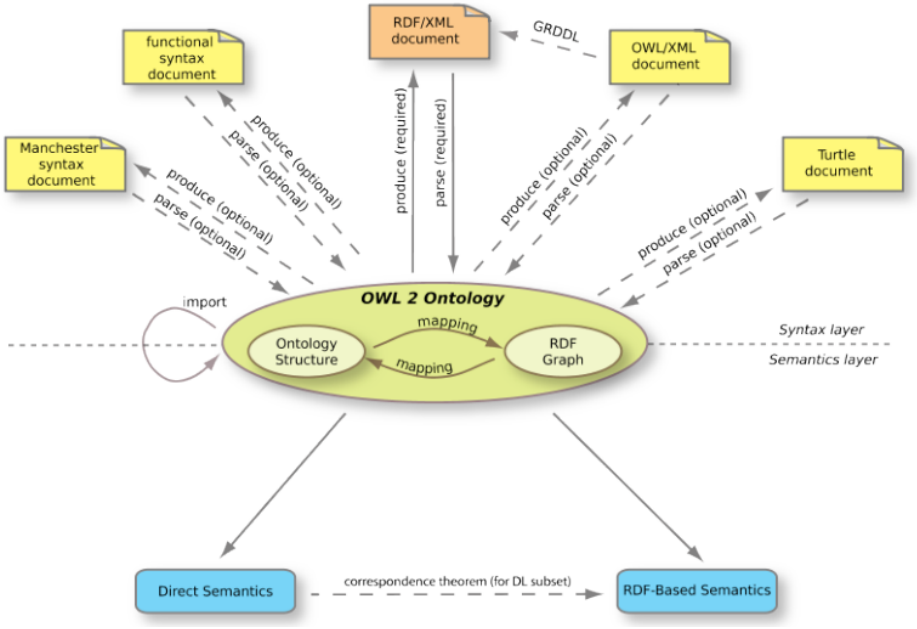

Figura 2.4: Estrutura de OWL2

Tal como as ontologias, OWL também apresenta os seus componentes básicos que são: classes, propriedades, instâncias de classes e relação entre instâncias. Um exemplo concreto da semântica usada em OWL onde se define o domínio e as propriedades [24]:

```
<owl:ObjectProperty rdf:ID="madeFromGrape">
  <rdfs:domain rdf:resource="#Wine"/>
  <rdfs:range rdf:resource="#WineGrape"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="course">
  <rdfs:domain rdf:resource="#Meal" />
  <rdfs:range rdf:resource="#MealCourse" />
</owl:ObjectProperty>
```

Neste exemplo, criou-se duas relações (objectProperties), "feito de uva"e "prato". Estas relação aplicam-se entre duas classes e possuem uma direção neste caso, o ponto de partida é o

domínio (*domain*) e o ponto de chegada é o alcance (*range*). Com isto em mente, verifica-se então que estas relações afirmam, respetivamente, o seguinte: "Vinho“ "feito de uva“ "Vinho de uva“ e "refeição“ "prato“ "refeição de prato“.

2.2.3 Metodologias

Dentro da comunidade de ciências de informação não existe um standard a seguir para construir uma ontologia. Contudo existem alguns artigos que caracterizam diferentes tipos de metodologias conforme a ontologia a aplicar. Segundo Gruninger e Lee existem três tipos de ontologias relativamente ao seu uso [25]:

- para comunicação;
- para inferência computacional;
- para reusabilidade do conhecimento.

O desenvolvimento de uma ontologia deve seguir ou ter em atenção as seguintes tarefas [26]:

- Selecionar o domínio e o âmbito;
- Considerar reusabilidade;
- Analisar os termos importantes;
- Definir classes e hierarquias;
- Definir propriedade de classes e restrições;
- Criar instâncias de classes.

2.2.4 Ferramentas

Ao longo dos últimos anos, têm vindo a crescer o número de ferramentas desenvolvidas para construção de ontologias. A primeira ferramenta desenvolvida foi no início dos anos noventas dado pelo nome de *The Ontolingua Server* pela Universidade de Stanford [27]. Nos últimos anos com o desenvolvimento da *Semantic Web* é possível encontrar na internet vários editores de ontologia, como Apollo, OntoStudio, Protégé, Swoop e TopBraid Composer. O artigo aqui referenciado compara todas as ferramentas e conclui, que dependo da finalidade da ontologia existem ferramentas melhores concebidas para tal efeito e outras em diferentes circunstâncias [28].

Protégé

Protégé é um editor de ontologias *open source* desenvolvido pela Universidade de Stanford. Oferece uma interface simples, intuitiva e gráfica para conceber ontologias. Este programa foi desenvolvido em Java e disponibiliza vários *plug-ins* que facilita o desenvolvimento de aplicações. Tem a capacidade de importar e exportar ontologias nos seguintes formatos: RDF/XML, Turtle, OWL/XML e OBO [29] [30].

2.2.5 Servidor de armazenamento triplo

Atualmente, existem vários servidores de armazenamento triplo. Os servidores apresentam diferentes tipos de armazenamento de esquemas. Uma grande parte dos destes sistemas concebidos para funcionar como base de dados relacional, possuem um sistema híbrido que gerir outras base de dados, ou até criar um esquema de tabelas SQL. O artigo que se segue revê todas as caraterísticas dos diversos servidores [31].

Virtuoso

Virtuoso é um sistema de acesso de dados pertencente a OpenLink que permite gerir base de dados relacionais e não relacionais. O servidor possui a capacidade de mapear base de dados não relacionais em relacionais [32]. Este servidor é *open source* e utiliza SPARQL como *query language*. É também o servidor utilizado pela *DBpedia* que manipula os dados da *wikipedia* bem como outras ontologias bem conhecidas través das suas ontologias, como se pode visualizar na figura 2.5 [33].

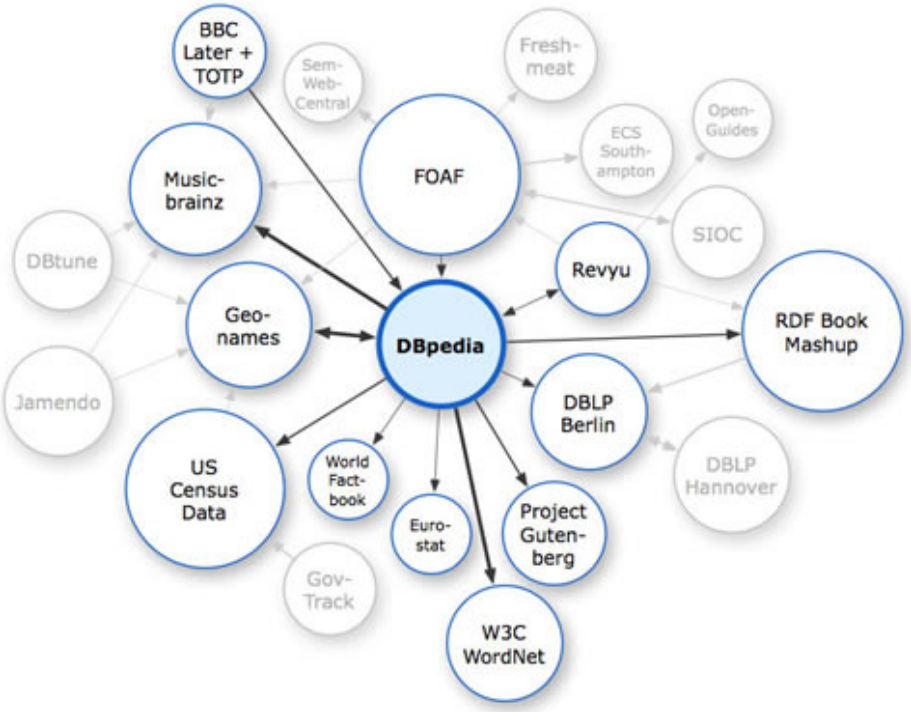

Figura 2.5: DBpedia - grafo de ontologias

Stardog

Stardog é uma base de dados de armazenamento triplo desenvolvido em Java pode escalar até 50 mil milhões de triplos. O servidor suporta por completo OWL2 e regras de inferência, contudo não permite mapear tabelas SQL para grafos RDF [34].

2.3 Conclusão

Este capítulo envolve toda a pesquisa realizada a priori, com foco especial em dois temas a gestão documental e a ontologia.

O estudo da gestão documental forneceu noções em duas vertentes, a nível da estruturação dos metadados e relativamente à sua importância no mundo empresarial.

Relativamente, a ontologia ficou definido pela Finantech a utilização do Protégé, na construção da ontologia, e o Virtuoso como o servidor responsável para a comunicação.

Capítulo 3

Modelação do Domínio

Ao longo deste capítulo será abordado a metodologia a utilizar durante o ciclo de vida do projeto e as etapas inicias realizadas a priori do desenvolvimento da aplicação. Para conceber um modelo teve-se em conta, em primeiro lugar, os requisitos propostos pela empresa e a situação atual no âmbito da gestão documental. Em segundo lugar, realizou-se junto com a empresa, um levamento dos tipos de documentos usados por departamento, com o objetivo de criar metadados que satisfaçam as necessidades da Finantech. Por fim utilizar a análise efetuada e aplicar o domínio em uma ontologia, que será a base de dados do sistema de gestão documental, AgiDoc.

3.1 Metodologia

Para este projeto considerou-se como metodologia mais assertiva a metodologia de protótipo por duas grandes razões. Primeiro, trata-se de uma prova de conceito, com um produto final que terá o objetivo de verificar a mais-valias da implementação de um sistema de gestão documental. Por outro lado, devido à grande necessidade de interações com o cliente ao longo do ciclo de vida do projeto, tanto na personalização da aplicação web como para refinar o estudo e a estruturação da ontologia no âmbito documental.

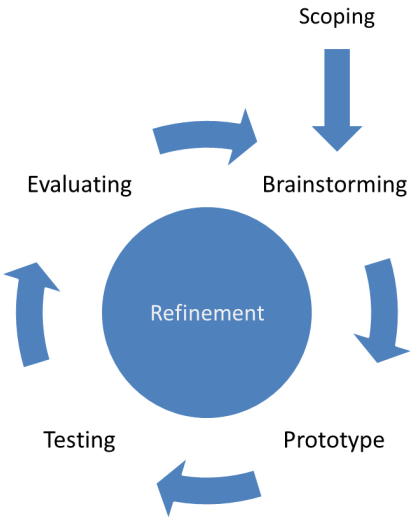

Figura 3.1: Fases da metodologia de prototipo

O diagrama 3.1 demonstra as diversas etapas essenciais do projeto, que foram mais do que uma vez executadas de forma a refinar o protótipo.

3.2 Especificação de requisitos

A primeira etapa concebida foi o levantamento dos requisitos. Para tal, realizou-se uma reunião com o administrador e o meu orientador da empresa. Na qual se esclareceu que, no presente, a Finantech não adota nenhum sistema de apoio à gestão documental, apenas existem regras ao nível dos diretórios para cada departamento. Aos olhos da Finantech, este projeto trata-se de uma prova de conceito, uma vez que pretendem consolidar estas tecnologias e conceitos no qual não trabalham no dia-a-dia.

Tendo isto em consideração enumerou-se um conjunto de requisitos pretendidos no produto final:

- Fundir e considerar os dados do SQL server na ontologia;
- Armazenar documentos no repositório;
- Classificar documentos segundo as necessidades da Finantech;
- Consultar o documento pela sua informação inerente;
- Desenvolver uma aplicação web;
- Seguimento das versões;
- Criar ecrãs capazes de consultar os documentos mais visualizados e criados pelo mesmo;
- Gestor de *check in* e *chek out*.

3.3 Estrutura e conceção

Passo seguinte, passa por criar uma arquitetura de projeto que satisfaça as condições e os objetivos da Finantech. A próxima imagem, figura 3.8, demonstra a arquitetura de alto nível desenvolvida que expõe aspetos fundamentais do conceito do sistema.

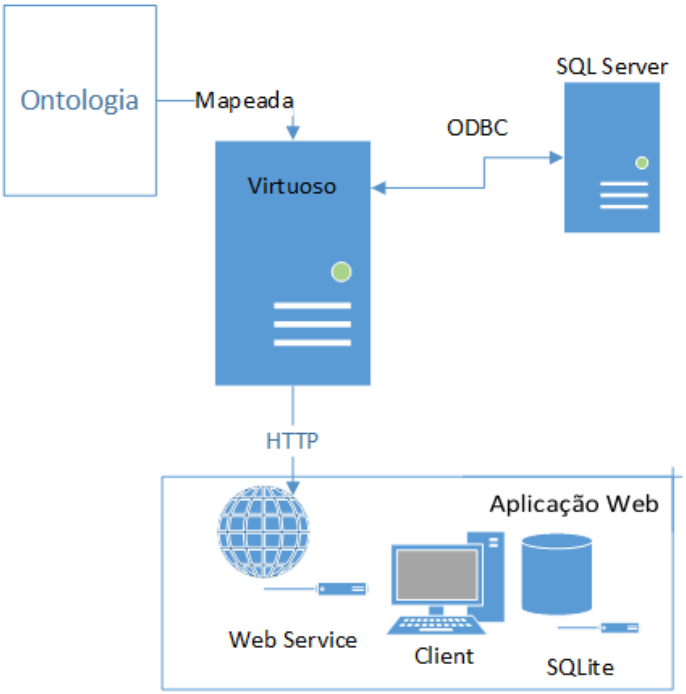

Figura 3.2: Arquitetura elaborada

Seguindo a arquitetura ilustrada anteriormente, iniciou-se por conectar por ODBC o SQL server ao Virtuoso. Tal ligação permitirá, numa primeira instância, criar uma ontologia base relativamente as tabelas de base de dados. O virtuoso mapeia, automaticamente, tanto as tabelas selecionadas bem como as suas próprias colunas.

Tudo isto é possível, graças à ligação ODBC que permite a atualização dos dados, em tempo real, entre o virtuoso e o servidor da Finantech. Basicamente, se for acrescentado ou modificado algum valor das tabelas, esse elemento será imediatamente atualizado na aplicação graças ao mapeamento efetuado pelo Virtuoso.

Com recurso à ontologia gerada pelo Virtuoso, manipulou-se a informação proveniente da mesma e expandiu-se o grafo e centralizou-se o domínio na documentação da Finantech.

No seguimento do mapeamento da ontologia no virtuoso, desenvolveu-se uma aplicação que trocará informação com o virtuoso por HTTP. A aplicação encontra-se dividida em duas camadas web service e Client. O web service é responsável de providenciar a API para o Client.

Os ficheiros ficarão alojados num repositório dentro do domínio da Finantech, onde será possível fazer download e upload de novos ficheiro.

Como este projeto é especifico para os trabalhadores da Finantech, faz todo sentido que o login seja efetuado nas máquinas na empresa pela sua conta no domínio da Finantech.

SQLite

Para cumprir os requisitos de *check in* e *check out* de ficheiros e de criação de ecrãs capazes de consultar os documentos mais visualizados e criados pelo mesmo, recorreu-se a um servidor SQLite que será implementado no back-end do Client. Com isto em mente, criou-se duas tabelas: FileStatus e UserRecord.

A tabela FileStatus (3.3) é responsável por atualizar se um determinado ficheiro se encontra disponível, *checked out* ou indisponível. O raciocínio por de traz da tabela é simples, quando existe uma interação com o ficheiro que envolva um dos estados, o sistema vai verificar em que estado (coluna status) se encontra o ficheiro dado pelo sua coluna fileId correspondente a data (coluna changeDate) mais recente, sabendo assim o seu estado atual.

Name	Type	Not Null	Default Value	Primary Key
id	INTEGER	1	null	1
changeDate	DATETIME	1	null	0
status	BOOL	1	null	0
userName	VARCHAR(50)	1	null	0
fileId	INTEGER	1	null	0

Figura 3.3: Tabela FileStatus

O ficheiro caso apresente o estado (*status*) a verdadeiro significa que se encontra disponível, ou seja, é possível fazer *download* do mesmo. Caso contrário verificam-se dois estados, o estado *checked out* que só se atribui ao utilizador que efetuou (coluna userName) o *download* e terá de realizar o *check in* do ficheiro para o mesmo ficar disponível. Para o resto dos utilizadores o ficheiro encontra-se no estado indisponível até o *upload* do ficheiro se verificar.

A tabela UserRecord (3.4) terá como objetivo acompanhar o histórico de documentos consultados, que permitirá estabelecer, na aplicação as vistas do próprio utilizador, os documentos mais consultados, query ordenada pela coluna count (coluna userName) e a vista dos documentos mais recentes, ordenando a query pela coluna lastViewDate.

Name	Type	Not Null	Default Value	Primary Key
id	INTEGER	1	null	1
objectId	INTEGER	1	null	0
objectType	INTEGER	1	null	0
userName	VARCHAR(50)	1	null	0
lastViewDate	DATETIME	1	null	0
count	INTEGER	1	1	0

Figura 3.4: Tabela UserRecord

Na conceção desta tabela ponderou-se acompanhar, para além dos documentos, outros tipos de objetos como ficheiros ou *collections*. Por isso existem as colunas objectType referente a um

dos objetos mencionados anteriormente e o objectId para identificar o objeto. Contudo, não se achou relevante seguir os outros objetos, daí o valor da coluna objectType ser fixo.

3.4 Elaboração da ontologia

Para conceber a ontologia utilizou-se uma ferramenta de abstração dada pelo nome de Protégé. Utilizou-se como ponto de partida a ontologia já gerada pelo Virtuoso depois de mapeado o SQL Server. O Protégé proporciona a funcionalidade de criar a partir da ontologia um grafo, o qual se pode vislumbrar na figura 3.5.

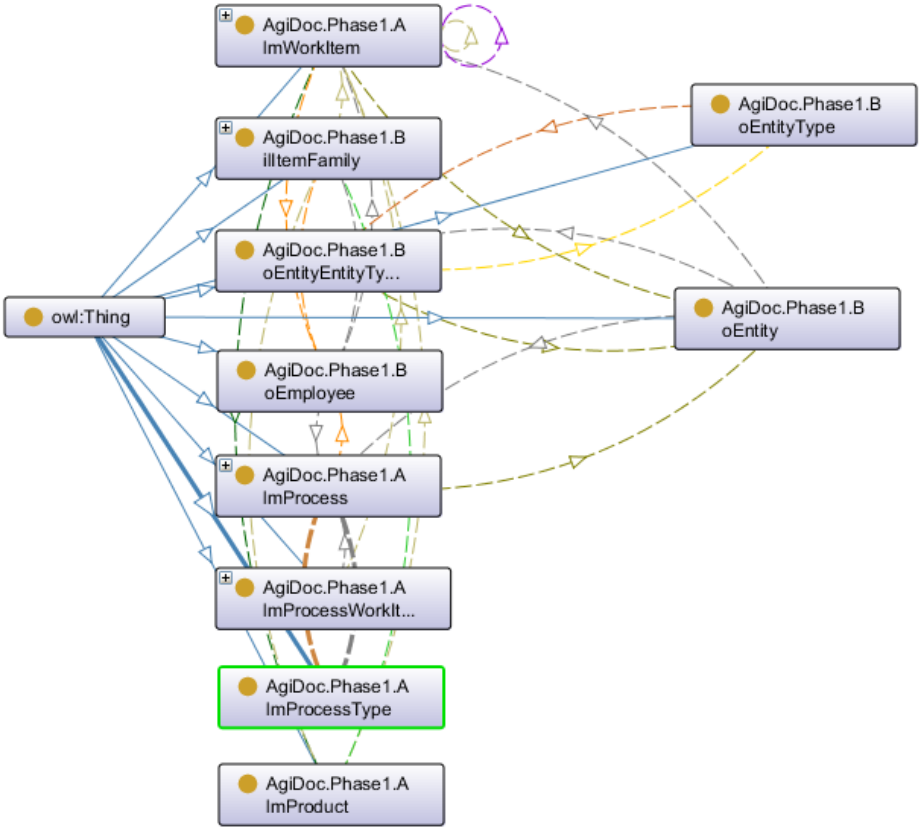

Figura 3.5: Ontologia em forma gráfica gerada pelo virtuoso

Cada tabela irá representar uma classe que possui diversas *data properties*, que representam as colunas e as *object properties* representam as relações entre as classes ontológicas. Desta forma, foi possível mapear as tabelas pretendidas do SQL Server para uma ontologia que será o ponto de partida. Posteriormente, esta ontologia será desenvolvida está atingir os padrões de coerência tanto da Finantech como para os objetivos da aplicação.

3.4.1 Classificação da documentação

Para que seja possível implementar um ontologia de domínio documental, é crucial classificar os documentos segundo informação que eles incidem. No caso da Finantech todos os dados correspondentes às entidades, produtos e entre outras informações importantes estão armazenadas em um servidor SQL.

Para o efeito, levantou-se as tabelas relevantes em conjunto com a empresa representadas na tabela 3.2, encontra-se no final do capítulo.

De seguida, investigou-se tipos de documentos que a empresa utiliza e tentou-se classificar os documentos através departamentos da Finantech (Class Groups) e cada Class Group possui uma lista de documentos tipos (Classes), por sua vez estes possuem metadados. O metadados estão divididos em três tipos: Metadados Genéricos, de Class e de Estrutura. A todos os documentos são lhes atribuídos metadados genéricos, os quais a Finantech classifica como metadados fundamentais encontram-se listados na tabela 3.1. Os metadados de Class são relativos à informação analisada da base de dados (tabela 3.2) e que permite distinguir as Classes, esta catalogação está representada na tabela 3.3. Relativamente aos metadados de estrutura serão relativos à conceção ontológica, simplesmente representa os *object properties* da ontologia.

Tabela 3.1: Metadados genéricos

Dados Genéricos	Tipo	Descrição
Author	String	Criador do objeto
Code	String	Código único gerado conforme os metados estruturais e versão que apresenta
CreateDate	DateType	Data de entrada do objeto no sistema
ID	Integer	Número identificador
ModifyDate	DateType	Data de modificação do objeto
Title	String	Título do objeto
Version	Integer	Número da versão
EditedBy	String	Autor da alteração do objeto
LatestVersion	Bool	Se é a última versão

A codificação (Code) existe em dois tipos de objetos: os documentos e ficheiros. Os documentos seguem a seguinte formatação D_[CG]_[C]_[ID]-Version na qual as siglas representam: D- Document, CD- Class Group, c- Class, ID- ID do documento. Os ficheiro apresentam a seguinte codificação F_[ID]-D_[D_ID]-Version nesta assinatura as siglas representam: F- File, ID- ID do ficheiro, D- Document, D_ID- ID do documento. O ID representa em qualquer dos casos representa o objeto inicial e por isso poderá não ser o mais recente.

Tabela 3.2: Informação a utilizar das tabelas SQL

Informação	Tabela	Observações	Query
Entidades	[BoEntity]	Todas as entidades	–Select para obter Entidades activas SELECT * FROM [bo].[BoEntity] WHERE [Active] = 1
	[BoEntityType]	Tipos de entidades	–Select para obter tipos das Entidades activas SELECT ET.[EntityTypeName], E.* FROM [bo].[BoEntity] E JOIN [bo].[BoEntityEntityType] EET ON E.[EntityId] = EET.[EntityId] JOIN [bo].[BoEntityType] ET ON EET.[EntityTypeId] = ET.[EntityTypeId] WHERE E.[Active] = 1 AND EET.[Active] = 1 ORDER BY E.[EntityId], ET.[EntityTypeName]
	[BoEntityEntityType]	Associação entre as entidades e os tipos. Uma entidade pode ter mais que um tipo.	
Famílias	[BilItemFamily]	Familias existentes na Finantech. A tabela também guarda as sub-familias. Uma row cujo campo [ItemFamilyParentId] é NULL representa uma família.	–Select para obter Familias ativas SELECT * FROM [bil].[BilItemFamily] WHERE [ItemFamilyParentId] IS NULL AND [Active] = 1
	[AlmProduct]	Tabela dos produtos. Está associada a família pelo campo [ItemFamilyId].	–Select para obter Produtos ativos de Familias ativas SELECT IFAM.[ItemFamilyName], P.* FROM [alm].[AlmProduct] P JOIN [bil].[BilItemFamily] IFAM ON P.[ItemFamilyId]= IFAM.[ItemFamilyId] WHERE IFAM.[ItemFamilyParentId] IS NULL AND IFAM.[Active] = 1 AND P.[Active] = 1
Projetos	[AlmWorkItem]	Tabela de Workitems	SELECT * newline FROM [alm].[AlmWorkItem] WI JOIN [alm].[AlmWorkItemType] WIT ON WI.[WorkItemTypeId] = WIT.[WorkItemTypeId] WHERE WIT.[WorkItemTypeName]='Opportunity' OR WIT.[WorkItemTypeName]='Proposal'
	[AlmWorkItemType]	Associação entre os processos e os workitems	
Colaboradores	[BoEmployee]	Todos os colaborades da Finantech	SELECT * FROM [AgiFoxDev].[bo].[BoEmployee]

Tabela 3.3: Metadados de Class

Class Group	Class	Propriedades	Opcional
Sales	Letter	Clients	
	E-mail	Clients	
	Presentation	Clients	
		Opportunity	Opcional
	Proposal	Clients	
		Opportunity	Opcional
	Client Order	Clients	
		Opportunity	
		Proposal	
	Contract	Clients	
Admin Services	Letter	Entities	
	E-mail	Entities	
	Work Contract	Colaborator	
Scrum Teams	Service Sheet	Clients	
	Product Description	Families	
		Products	
	Script	Families	
		Products	
	Requirements	Families	
		Products	
Help Desk	Service Sheet	Clients	
	User Manual	Families	
		Products	Opcional
	Script	Families	
		Products	

Importante salientar que, por exemplo, um documento do tipo e-mail pode ter mais do que um cliente associado, bem como mais que um ficheiro associado, neste caso seria um conjunto de e-mails sobre um determinado assunto que envolvia o mesmo cliente.

A partir da análise aqui efetuada, representou-se tudo num diagrama de classes 3.6, onde é possível visualizar as relações, propriedade e atributos das classes que facilitará a implementação da ontologia.

De modo a garantir uma maior flexibilidade, adicionou-se outro Class Group chamado Genérico, o qual possui todos os documentos tipos ilustrada na tabela 3.3. Esta class Group foi proposta pela empresa, que argumentaram que seria impossível prever todos os documentos que cada departamento utiliza ou cria nos próximos anos.

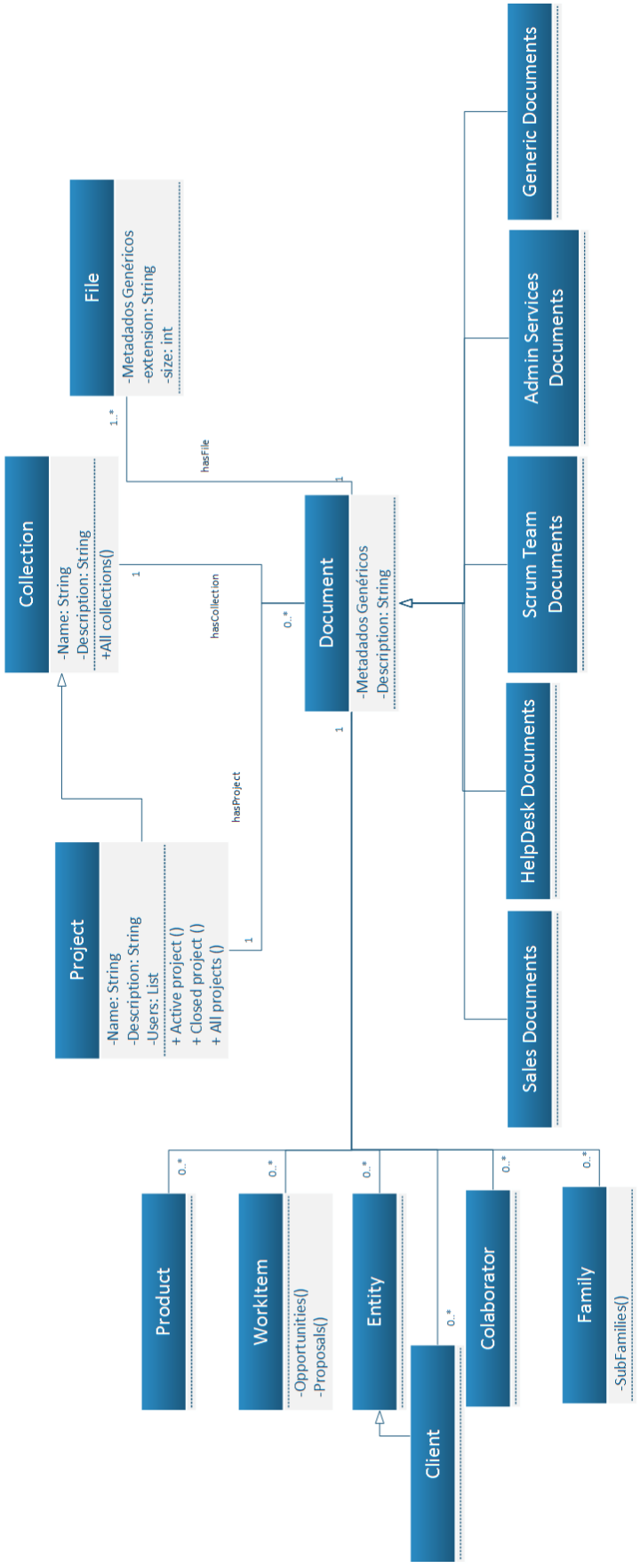

Figura 3.6: Diagrama de classes

3.4.2 Ontologia

Elaboração das Classes

No diagrama de classes, figura 3.6, o documento é composto por várias subclasses, as Class Group. Cada Class Group é constituído por várias subclasses, as Class, conforme a tabela 3.3.

Neste domínio, o documento representa o conjunto de informação, relativo às propriedades e metadados inerentes aos ficheiros que este agrega. Sublinha-se que um documento pode ser constituído por um ou mais ficheiros conforme a necessidade. Ficheiro representa o recurso em formato digital, como a localização no repositório, a extensão e o tamanho. Transpondo para um caso prático, um documento do tipo Proposal deverá ter um ficheiro *excel* com o valor descriminado da proposta e o ficheiro *word* com os detalhes da proposta.

As Collections funcionam como agrupadores de documentos e tem o objetivo de juntar documentos caso existe necessidade para tal. Os projetos partilham as mesmas funcionalidades mas é restrito para uma lista de utilizadores e que estejam a desenvolver um determinado projeto.

As classes como Entity, WorkItem, Collaborator, family e Product são relativas às tabelas da base de dados da Finantech e têm o propósito de caracterizar os tipos de documentação de forma a concretizar a lógica apresentada na tabela 3.3. Importante referir que a tabela WorkItem é constituida por Proposals e Opportunities contudo estes selects serão efetuados na API.

A partir da ontologia gerada pelo Virtuoso estabeleceu-se equivalências com classes criadas no Protege.

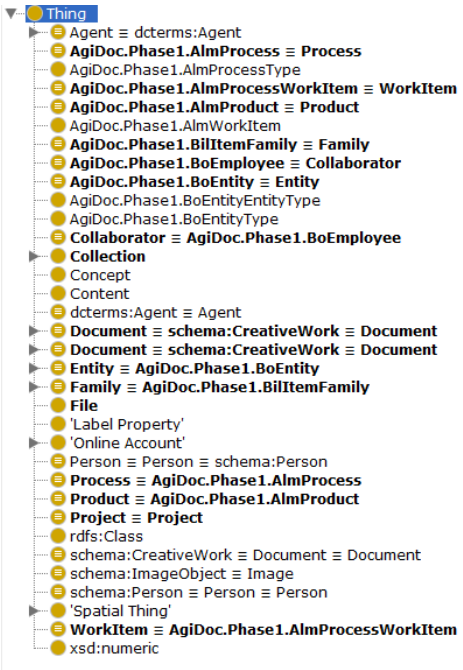

Figura 3.7: Classes da Ontologia

Ao longo da sua conceção, procurou-se estabelecer conceitos de outras ontologias, como FOAF e SKOS. Nas quais se reutilizou as definições de project (FOAF) e collection (SKOS).

Aplicação das propriedades

Na ferramenta Protege existem dois tipos de propriedade, as *Data Properties* e as *Object Properties*. As *Object Properties* destinam se a conceber relações identificadas no class diagram, basicamente, são relações que nos permitem chegar por queries as *Data Properties* pretendidas. As *Data Properties* representam e caracterizam (conforme o tipo) o conjunto de dados associados a uma classe e numa perspetiva prática representa informação de base de dados.

Finalizada esta fase é só adicionar a ontologia ao Virtuoso como Schema e é possível avançar para implementação da aplicação.

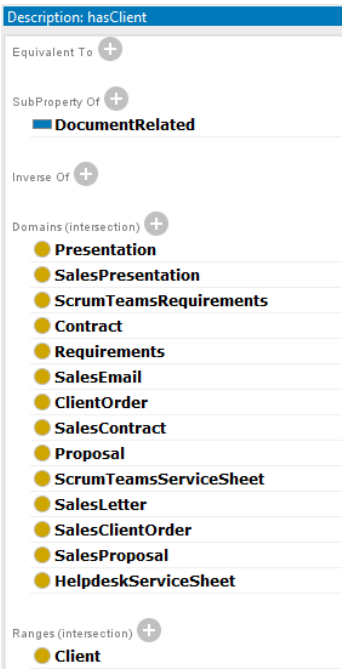

Figura 3.8: Exemplo da propriedade hasClient

3.5 Conclusão

Ao longo deste capítulo preparou-se todo o modelo e método de trabalho sobre o qual será desenvolvido o sistema. Tratou-se de caracterizar todo o domínio da documentação na Finantech e todos os dados importantes sobre o qual se enquadra a documentação.

Capítulo 4

Desenvolvimento da aplicação web

A aplicação web foi desenvolvida em .NET na linguagem C# a pedido da empresa, uma vez que é a plataforma que mais utilizam devido a parceria com a Microsoft. A aplicação segue a estrutura MVC em ASP.NET e encontra-se dividida em duas camadas, web service e client. Optou-se na separação com o intuito de facilitar a possibilidade de implementação de novos clientes, reusabilidade e a capacidade de funcionar como *black box*, ou seja, poderá ser desenvolvida por outras pessoas sem conhecimentos em SPARQL, em trabalhos futuros.

A aplicação pretende satisfazer os seguintes objetivos:

- criar documentos e adicionar ficheiros 4.2.2;
- editar os metadados dos documentos 4.2.2;
- consultar documentos pela sua informação relativa aos metadados 4.2.2;
- disponibilizar ecrãs sobre os documentos mais visualizados e vistos mais recentemente pelo próprio utilizador 4.2.2;
- histórico de versão do documento e ficheiro 4.2.2;
- visualizar os documentos criados pelo o utilizador 4.2.2;
- acompanhar os projetos em que o utilizador participa 4.2.2;
- alertar ficheiros que se encontram *checked out* 4.2.2.

4.1 Web service

Antes de avançar para a codificação do web service, planeou-se quais os componentes necessários para estabelecer ligação com a base de dados e os procedimentos de segurança. O Virtuoso disponibiliza drivers ADO.NET, o que garante a comunicação ODBC para .NET *Framework* e

permite fazer pedidos ao servidor sem ter de lidar com protocolos. A segurança não se achou prioritário, visto que se trata de um protótipo de acesso apenas local.

No desenvolvimento do web service adotou-se a estrutura MVC, mesmo não apresente qualquer *view* mas permitia uma melhor organização e reutilização do código para a camada Client, no caso dos modelos.

De forma a cumprir os objetivos anteriormente identificados, desenvolveu-se um conjunto de classes capazes de satisfazer os requisitos pretendidos no Client, ilustrado na figura 4.1. De salientar que o bloco Finantech Data não é uma classe, mas como as classes inseridas no bloco apresentam quase todas os mesmo métodos torna-se mais fácil de se perceber e representar. Outro detalhe importante, a classe collection possui os métodos da subclasse project.

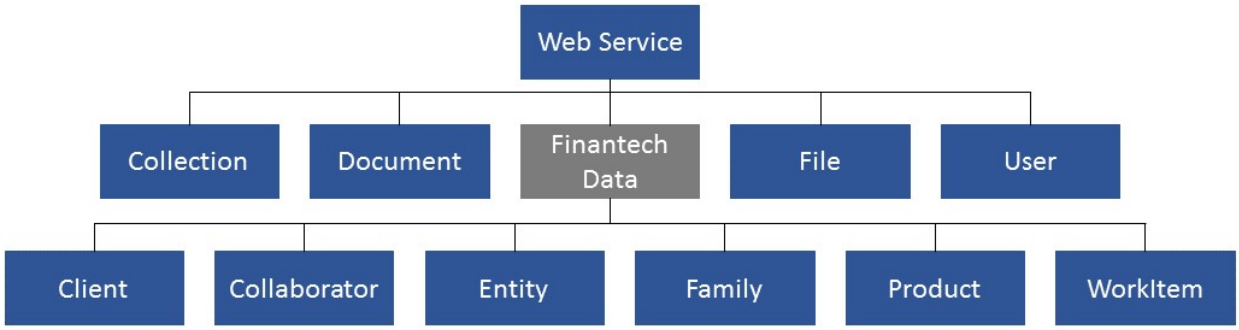

Figura 4.1: Classes do Web service

De seguida, analisou-se os dados relevantes que cada classe deveria conter. O ASP.NET permite criar modelos onde se identificam os *data members* que caraterizam as classes e subclasses importantes para o sistema. Em baixo ilustra-se um exemplo de modelo, neste caso o modelo da classe Collection que possui 6 *data members*. De referir, que numa ontologia os identificadores são IRIs, que direcionam a um endereço que permite ver os seus atributos e relações. A collection poderá conter documentos, esses documentos serão identificados numa lista de *strings* com o nome de DocumentsIris. Os restantes *data members* são meramente informativos.

```
public class Collection
    {
        public string Iri { get; set; }
        public string Name { get; set; }
        public string Description { get; set; }
        public DateTime CreationDate { get; set; }
        public string Author { get; set; }
        public List<string> DocumentsIris { get; set; }
    }
```

SPARQL

Como a base de dados é relacional, não existem variáveis do tipo *serial*, o que nos dificulta a adição de novos objetos.

Para adicionar, por exemplo, um novo ficheiro é necessário realizar duas *queries* em SPARQL a primeira para identificar qual é o id já utilizado e depois sim pode-se adicionar o ficheiro à base de dados.

```
Query para saber o ultimo ID  -> sparql
SELECT ?id FROM <{ GlobalStrings.AgidocGraph }>
WHERE {{?file a agidoc:File . ?file agidoc:hasID ?id}}
ORDER BY DESC(?id) LIMIT 1

Query para adcionar novo ficheiro -> sparql INSERT DATA
{{GRAPH <{ GlobalStrings.AgidocGraph }>{{agidoc:{ code } a agidoc:File .
agidoc:{ code } agidoc:isFileOf <{ documentIri }> .
<{ documentIri }> agidoc:hasFile agidoc:{ code } .
agidoc:{ code } agidoc:hasCreateDate \"
{ creationDate.ToString("yyyy'-'MM'-'dd'T'HH':'mm':'ss'.'fff") }\" .
agidoc:{ code } agidoc:hasTitle \"{ title }\" .
agidoc:{ code } agidoc:hasAuthor \"{ author }\" .
agidoc:{ code } agidoc:hasID { id } .
agidoc:{ code } agidoc:hasVersion { version } .
agidoc:{ code } agidoc:isLatestVersion 1 . agidoc:{ code } agidoc:hasAddress
\"{ Regex.Escape(address) }\" . agidoc:{ code } agidoc:hasExtension
\"{ extension }\" . agidoc:{ code } agidoc:hasSize { size } }}
```

As diferenças entre SQL e SPARQL é que em SQL é necessário saber as colunas das tabelas em SPARQL necessário conhecer as propriedades dos objeto. Por isso a query de adição é relativa a um ou mais grafos neste caso é só ao Agidoc e é necessário definir todas as propriedades, ou seja as *data properties* e as *object properties*. Ao contrário das bases de dados não relacionais, no exemplo da adição, caso seja esquecida uma propriedade não dá origem a um erro simplesmente não fica atribuída. Por isso é importante criar modelos com todas a propriedades de cada objeto para minimizar os erros.

API

Na tabela 4.1 encontram-se listados alguns dos métodos criados de cada classe. Como seriam muitos métodos omitiram-se os métodos repetidos onde a diferença seria só o objeto, pois estes possuem as mesmas *queries*. Substituir classFinantechData por uma Classe inserida no bloco Finantech Data.

Tabela 4.1: API desenvolvida

Classes	Rota	Descrição
Collection	GET api/collections/{name}	Lista de documentos associados a uma determinada collection
	PUT api/collections/projects/{name}/user	Insere novo utilizador a um projeto
Document	GET api/documents/all	Lista todos os documentos existentes
	POST api/documents/new	Adiciona um novo documento
	GET api/documents/{code}	Retorna os metadados do documento, caso exista o documento com o código code
	GET api/documents/{code}/history	Retorna o histórico de versões do documento
	PUT api/documents/{code}	Edita os metadados do documento
	DELETE api/documents/{code}	Apagar o documento
	GET api/documents/{id}/files	Lista de ficheiros do documento com o id id
	GET api/documents/{descriptors}/ /properties	Lista de propriedades de uma ClassGroup
	GET api/documents/groups	Lista de todas as ClassGroups existentes
	GET api/documents/{group_d}/types	Lista de todas as Classes de um determinado ClassGroup
	GET api/documents/list	Lista de todos os documentos e suas propriedades
File	GET api/files/{id}/history	Retorna o histórico de versões do ficheiro
	POST api/files/upload	Adiciona o ficheiro no repositório
	GET api/files/code/extension	Retorna a extensão do documento
Finantech Data	GET api/classFinantechData	Lista todos os elementos relativos a uma classe
	GET api/classFinantechData/id	Retorna elemento e os respetivos dados da classe
	GET api/classFinantechData/{id}/docs	Lista todos os documentos de um elemento da classe Finantech Data (substituindo por classFinantechData)
User	GET api/user/{user}/documents	Lista de documentos criados pelo utilizador em causa
	GET api/user/user/projects	Lista de projetos em que o utilizado está inserido

De notar, que as rotas da API retornam informação em JSON. O próximo passo é criar um Client capaz de interpretar e dispor a informação em JSON proveniente do web service.

4.2 Client

O Client representa toda a camada responsável por gerir as interações com o cliente e o tratamento da informação do web service de modo a apresentar uma interface amigável para o utilizador comum. A capacidade de dispor a informação de forma organizada, intuitiva e percetível um fator crucial e deve ser o pilar do desenvolvimento.

Como já foi mencionado adotou-se a estrutura MVC, na qual se reutilizou os mesmos modelos do web service e conforme as vistas criavam-se modelos específicos para as vistas, dado pelo nome de ViewModel.

A maior parte das vistas apresentam duas camadas, a camada HTML e de JavaScript. A camada de HTML foi codificada maioritariamente em Razor, permite uma implementação mais prática e dinâmica, uma das mais valias do C#. A camada de JavaScript foi desenvolvida para manipular as páginas no lado do cliente para garantir páginas dinâmicas.

4.2.1 Back-end

Ao nível dos controladores, foi preciso tratar as informação de JSON para os modelos convenientes ou enviar simplesmente o objeto para a vista ou no caso de adição ou edição realizar o tratamento contrário.

Dentro dos objetivos previstos no início do capitulo, falta contemplar a contabilização do número de visualizações e o estado dos ficheiros implementou-se um servidor SQLite, algo simples para estabelecer uma comunicação com o Client. Isto deveu-se ao fato de não existir acesso a lista de utilizadores do domínio Finantech (existe acesso à tabela employee mas não fornece os utilizadores do domínio), para rever as tabelas e o raciocínio aplicado 3.3.

4.2.2 Front-end

De forma a estruturar a aplicação concebeu-se um esquema de páginas, figura 4.2, de referir existema várias páginas de pesquisa de documentos conforme o tipo de documento que o utilizador procura.

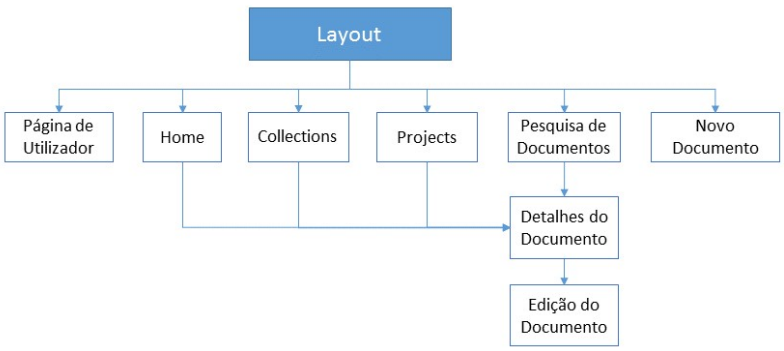

Figura 4.2: Esquema de páginas

Referente a figura 4.2, é importante esclarecer que as páginas Home, Collections e Projects possuem documentos associados e quando se clica num deles é-se direcionado para a página de Detalhes do Documento.

Perante este conjunto de conteúdos estabeleceu-se um *layout* global, figura 4.3, que contém uma barra de navegação no topo da página e outra barra de navegação lateral relativa ao utilizador *logged in*. Na barra de navegação superior encontram-se *links* para as diferentes páginas existentes e uma caixa de texto que permite pesquisar uma *string* no grafo. A barra lateral é relativa ao utilizador com 3 *links* que mostram informação própria, como os documentos que criou, os projetos no qual está inserido e o documentos por fazer *check in*. Esta barra dispõe ainda de um botão que permite criar um documento novo. O retângulo a vermelho representa onde o conteúdo das páginas será exposto.

Figura 4.3: *Layout* do Client

Ao longo desta secção iremos abordar os ecrãs disponíveis nestes conteúdos e os procedimentos relevantes. Alerta-se o facto de que o próximos ecrãs não apresentam o *layout* para diminuir o tamanho da imagem.

Home

Home representa a página *default* da aplicação, nesta página apresenta 3 ecrãs relativos ao utilizador *logged in*, como se pode ver a figura 4.4. O primeiro lista os documentos criados pelo utilizador. O segundo lista os documentos consultados mais recentemente. Por fim, o terceiro demonstra os documentos consultados com mais frequência.

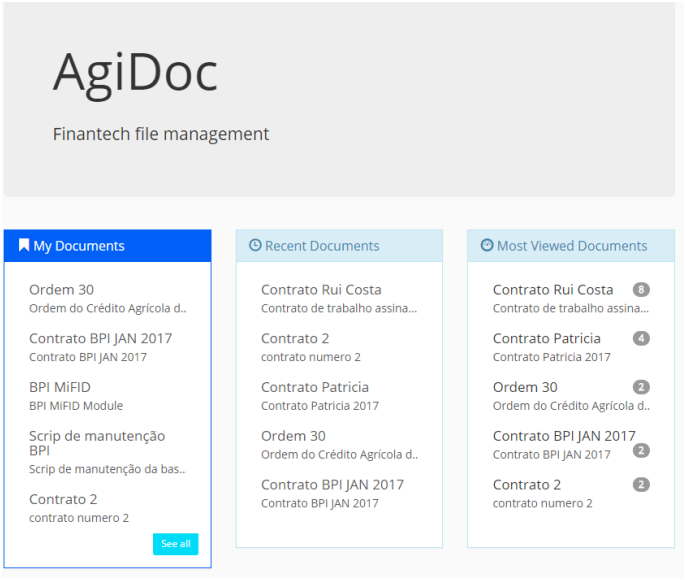

Figura 4.4: Disposição da página Home

Página de utilizador

A página de utilizador acede-se a partir da barra lateral e apresenta um ecrã com 3 *tabs*, como demonstra a figura 4.5. A *tab* de My Documents apresenta todos os documentos criados pelo utilizador. A *tab* de Project mostra os projetos no qual o utilizador está inserido. A última *tab* dada pelo nome de Checked-out Files mostra os ficheiros que foram descarregados do sistema para modificar o conteúdo. Enquanto os ficheiros Checked-out não forem repostos ninguém terá acesso à versão *downloaded*.

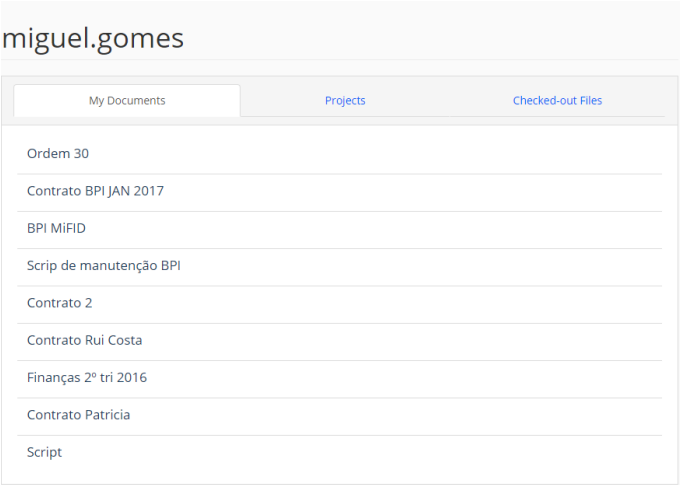

Figura 4.5: Disposição da página de utilizador

Projects e Collections

A página Project dispõe todos os projetos existentes em módulos, como se pode ver na figura 4.10. A página encontra-se com a opção de criar novo projeto aberto, por isso aparecem os campos obrigatórios. A página Collections é exatamente igual à página Projects.

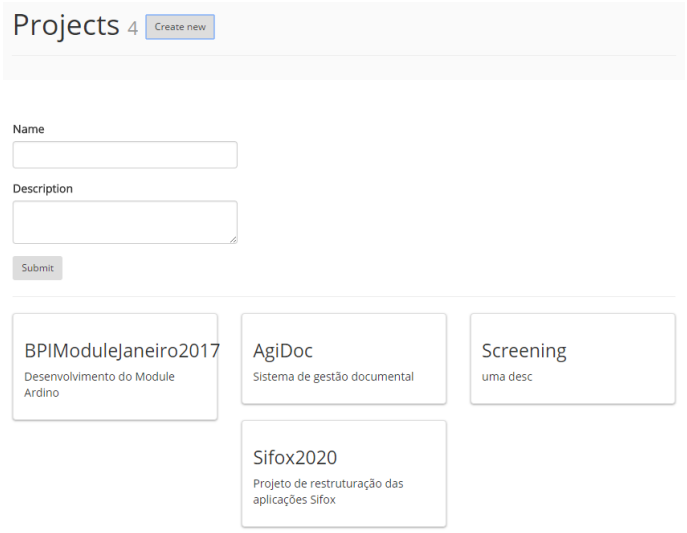

Figura 4.6: Disposição da página Projects

Ao clicar num dos projetos, somos direcionados para uma outra vista, onde é possível apagar ou editar o mesmo. Para além disso, nessa vista também permite ver o estado atual do projeto (ativo ou finalizado) e os utilizadores inseridos no projeto e ainda acrescentar novos ou eliminar algum dos existentes. Ao clicar numa *collection* acontece exatamente a mesma coisa, só que as *collections* não têm utilizadores ou estados associados.

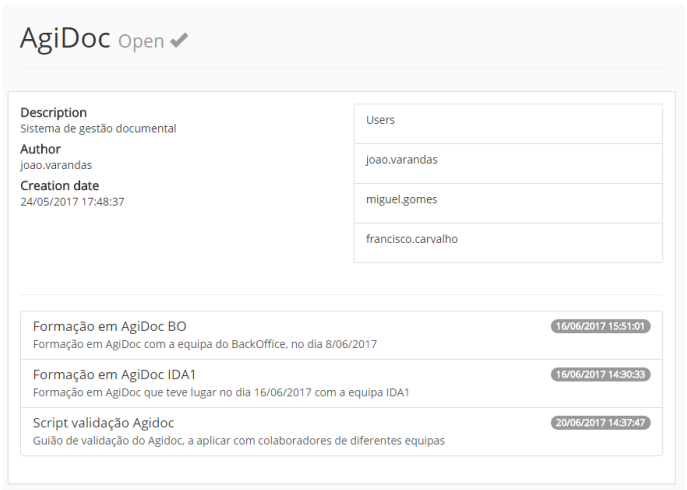

Figura 4.7: Detalhes de um projeto

Novo Documento e Edição do Documento

Um documento apresenta obrigatoriamente um grupo e tipo, a combinação é garantidamente correta, uma vez que o tipo é gerado dinamicamente. Está combinação possui propriedades obrigatórias ou funcionais. Para além disso, todos os documentos são constituídos por um título e uma descrição, a descrição tem um mínimo de palavras para garantir a sua devida contextualização. Por fim, eles podem estas associados a um *Project* ou *Collection*.

O Novo Documento é um formulário inserido num *modal*, espécie de um *pop up*. O formulário segue a classificação ontológica e quando é preenchido um grupo (*ClassGroup*) preenche o *dropdown* do tipo (*Class*) com os respetivos tipos do grupo. Como cada combinação de grupo e tipo possuem propriedades, por isso quando ambas encontram-se preenchidas o *modal* cria dinamicamente campos ao formulário das propriedades associadas. No exemplo demonstrado na figura 4.8 o documento que se pretende adicionar pertence ao grupo *Admin Services* do tipo *Work Contract* e só uma propriedade associada que é *Collaborator* de cariz obrigatório.

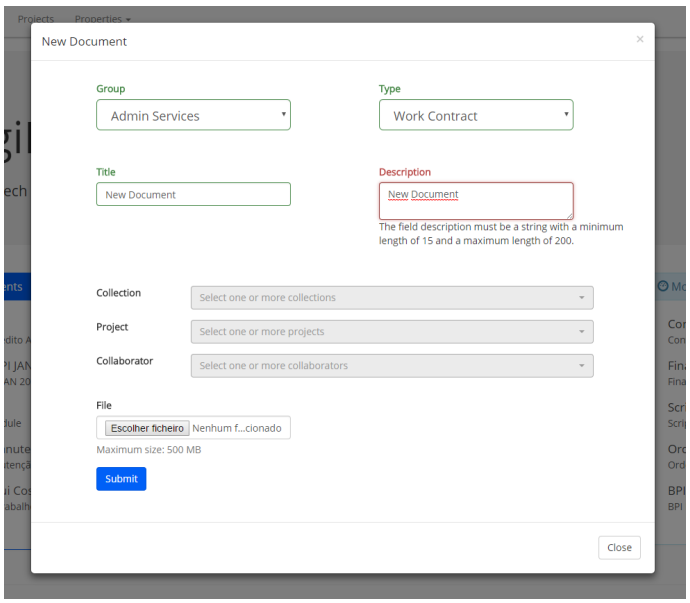

Figura 4.8: Formulário New Document

Como se pode identificar à medida que são preenchidos os campos, eles serão validados, a verde, ou inválidos a vermelho. Caso exista casos inválidos o *Client* não executa o botão *Submit*.

A Edição do segue exatamente os mesmos princípios a única diferença é que o formulário já se encontra com campos preenchidos.

Pesquisa de Documentos

Na aplicação existem três tipos de páginas de pesquisa documental: pesquisa por data, por departamento e por propriedades.

A página de pesquisa *By Date* apresenta em caixas expansíveis de todos os anos que possuem documentos. Quando se clica num desses anos a caixa expande apresentará todos os meses desse ano que têm documentos associados. Por sua vez, ao clicar num desses meses o a caixa também expande e é listado todos os documentos relativos ao mês do tal ano.

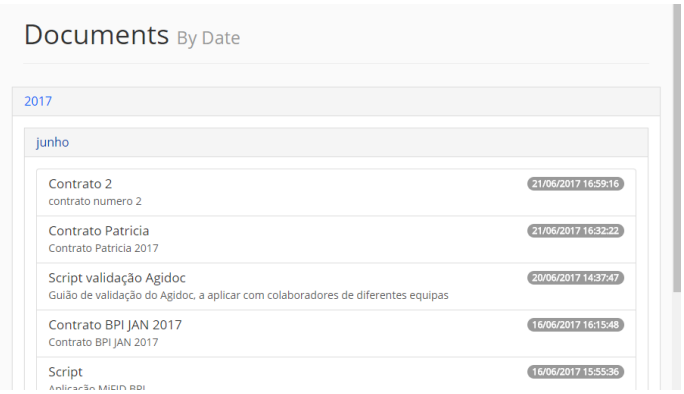

Figura 4.9: Disposição da página By Date

A página de pesquisa *By Group* permite ao utilizador consultar documentos por departamento e filtrar os documentos pelos tipos existentes em cada. De salientar que o *drop down* é preenchido automaticamente pelos tipos existentes dentro do departamento.

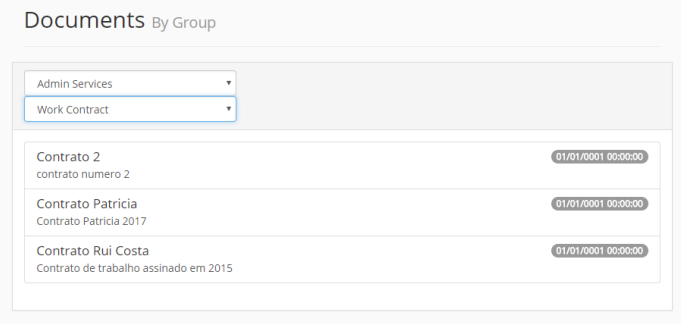

Figura 4.10: Disposição da página By Group

Existe mais do que uma página de pesquisa por *Properties*, por isso, a *navbar* é constituída por um *dropdown*. A página, figura 4.11, é constituída por um *dropdown* o qual permite escolher um elemento da propriedade da lista completa. Quando selecionado um elemento a *datatable* é dinamicamente preenchida com os documentos que possuem essa propriedade. A *datatable* é um *plug-in* para jQuery e permite por pesquisar por um elemento da tabela e de ordenar alfabeticamente ou numericamente as colunas.

Figura 4.11: Disposição da página Properties

Ao clicar num elemento ativa-se uma vista que mostra os dados respetivos do documento selecionado, ainda permite editar ou apagar documento ou adicionar um novo ficheiro. A página adquire a seguinte disposição, figura 4.12.

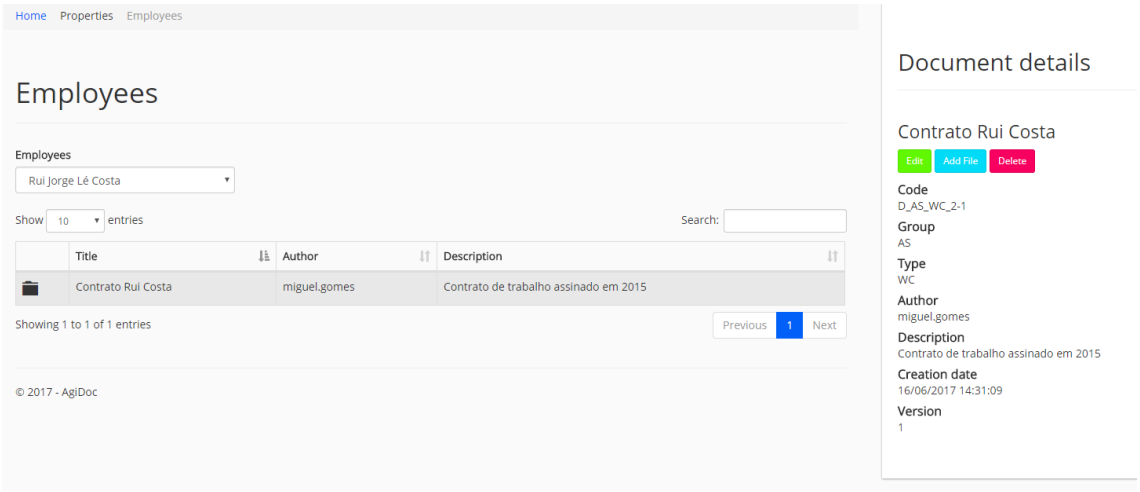

Figura 4.12: Disposição da página Properties com detalhes

Detalhes do documento

Esta vista é possível de aceder ao clicar num objecto documento a partir de qualquer outra página, como prevê o esquema da figura 4.2. Nesta vista é possível, para além de ver alguns metadados do documento, ver o histórico de versões (botão History), ir para a vista de edição do documento (botão Edit), adicionar um novo ficheiro, descarregar a versão mais recente do ficheiro (check out lastest version) ou ver o histórico do ficheiro e descarregar uma outra versão ao carregar no ficheiro, como exemplificado na figura 4.13.

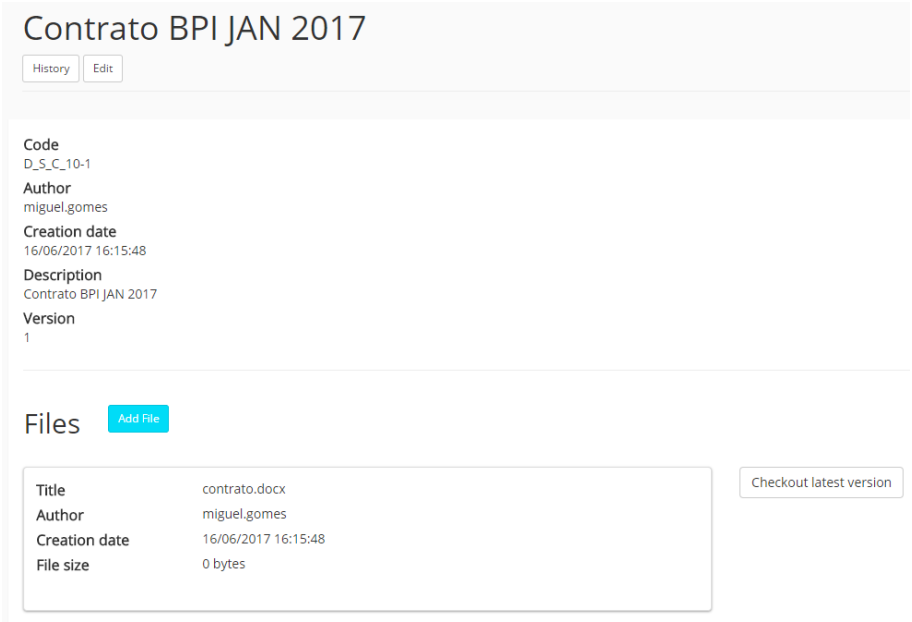

Figura 4.13: Disposição da página de Detalhes do documento

A vista de histórico do ficheiro, 4.14 é importante referir que quando alguém faz *Checkout* de um ficheiro, qualquer que seja a versão, faz com que ele fique indisponível para outros utilizadores. Para o utilizador que baixou o ficheiro fica no estado de *Checked out file* que permite fazer *upload* do ficheiro. Sempre que se faz *upload* de um ficheiro ele passa a ser a versão mais recente e só depois fica disponível para todos os outros utilizadores.

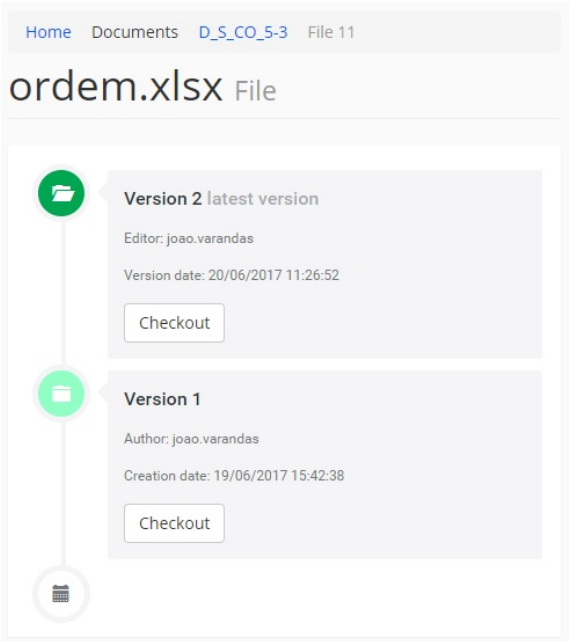

Figura 4.14: Histórico de um ficheiro

4.3 Conclusão

Este capítulo demonstra como foi implementada a aplicação dado os requisitos atribuídos pela Finantech.

Ao longo do desenvolvimento da aplicação manteve-se o cuidado de estruturar o código segundo a *framework* MVC, para facilitar a possível expansão do protótipo por mais elementos da Finantech.

Capítulo 5

Validação

Para validar o sistema desenvolvido, optou-se por envolver os pessoal da Finantech. Já que futuramente, em teoria seriam eles que utilizariam o sistema todos os dias, por isso faz todo o sentido que sejam eles a validar o sistema.

Perante isto, o plano de validação adotado, passou por convidar dois trabalhador de cada departamento. Seria importante que houvesse colaboradores de todos os departamentos porque cada departamento possui diferente métodos de trabalho e formas de arquivamento de documentos. Só assim a validação seria coerente.

De modo, que fosse possível avaliar o sistema uma rotina. A rotina estava dividida em três partes, a primeira parte demonstrou-se o levantamento de problemas encontrado se perguntou-se se concordavam ou se acrescentavam mais algum. Na segunda parte, realizou-se uma demonstração das capacidades do sistema desenvolvido. Por fim, pediu-se para avaliar de 1 a 5, em que o 1 seria avaliação mais negativa e 5 o mais positivo, quanto a dois fatores, resolução dos problemas antigos e quanto à qualidade da aplicação como representa a tabela 5.2.

Os problemas encontrados foram quatro relativos a gestão de documentação atual que são:

1. Dificuldade em encontrar documentação;
2. Duplicação de documentos dentro de todo o domínio;
3. Constante desalinhamento ou falta de versões;
4. Perda de documentação.

Tabela 5.1: Parâmetros a avaliar

Fatores	Parâmetros
Melhorias	1.1- Dificuldade em encontrar documentação
	1.2- Duplicação de documentos dentro de todo o domínio
	1.3- Constante desalinhamento ou falta de versões
	1.4- Perda de documentação
Aplicação	2.1- Qualidade da interface
	2.2- Organização da aplicação

5.1 Resultados

Os resultados têm por base a avaliação efetuada pelos colaboradores da empresa, uma amostra de oito pessoas, duas de cada de departamento.

Relativamente aos resultados da primeira parte da rotina, todos concordaram que, atualmente na Finantech, existiam esses problemas e para além desses ninguém acrescentou outro defeito para além dos já identificados.

A terceira parte, a avaliação propriamente dita, obtiveram-se os seguintes resultados:

Tabela 5.2: Avaliação dos colaboradores

Departamento	Parâmetros [1-5]					
	1.1	1.2	1.3	1.4	2.1	2.2
Admin Services	5	5	5	5	5	4
	5	5	5	5	4	4
Help Desk	5	5	4	5	4	4
	5	5	5	5	4	5
Sales	5	5	5	5	5	5
	5	5	5	5	3	4
Scrum Teams	5	5	4	5	4	5
	4	5	5	5	4	5
Finantech (média)	4.9	5	4.8	5	4.1	4.5

A tabela 5.2 demonstra as avaliações atribuídas, entre 1 a 5, aos parâmetros (colunas) apresentados na tabela 5.1, por dois colaboradores de cada departamento (linhas).

5.2 Análise e discussão

Uma vez que a Finantech não possui nenhum sistema de gestão documental já se esperava que as avaliações tivessem um cariz positivo. Recebeu-se *feedback* sobre certas melhorias para implementar futuramente. Uma das melhorias que a maior parte gostaria que a aplicação possuísse uma vista que fosse capaz de pesquisar o documento com todos os metadados disponíveis do sistema e retirar a pesquisa por diferentes páginas.

Outra que mais que um colaborador apontou, estava relacionada com as versões. Eles gostariam que as versões fossem dispostas em forma de árvore, por exemplo, a versão 2 é uma atualização da versão 1, mas a versão 3 pode ser outra atualização da versão 1 ou 2. Um esquema em forma de árvore permite concluir as origens das versões, em vez da atual, que dispõe as versões da mais antiga para a mais recente.

5.3 Conclusão

Todos os colaboradores consideram o sistema extremamente útil no seu dia-a-dia e gostariam que fosse implementado na empresa. Muitos deles afirmam gastar muito tempo à procura de um certo documento, entretanto desiste e envia um e-mail a pessoa que guardou o documento ou

que sabe que possui. Todo este tempo envolve custos para a empresa e causa "dores de cabeça" desnecessárias aos colaboradores.

Capítulo 6

Conclusão

Ao longo deste percurso de trabalho na Finantech, foi desenvolvido um sistema de gestão documental, para tal foi necessário usar tecnologias que a empresa não utiliza no seu dia-a-dia.

O desenvolvimento do sistema dividiu-se em três fases, Modelação do Domínio, Desenvolvimento da aplicação web e a validação.

Durante a fase de Modelação, houve vários problemas, como o tempo consumido para implementar a ontologia no Virtuoso com as tabelas dos servidores da Finantech, a análise dos tabelas de base de dados e as várias interações com os colaboradores da Finantech levou a refinamentos na ontologia.

O desenvolvimento da aplicação *web*, devido ao tempo consumido na modelação, foi encurtado mas foi possível cumprir todos os requisitos propostos.

A validação junto dos colaboradores da Finantech, permitiu avaliar o protótipo e perceber gostariam que fosse implementado. A avaliação foi unânime e todos os colaboradores dos diferentes departamentos gostariam de ver o sistema de gestão documental nas suas máquinas.

Os requisitos que a empresa propôs foram cumpridos e implementação deste sistema trará, sem dúvida nenhuma, vantagens para a empresa em termos de:

- Organização sistemática da documentação;
- Redução do tempo de procura da documentação;
- Histórico e versionamento da documentação.

Agora resta a Finantech analisar as mais valias e custos associados relativos a esta prova de conceito aqui desenvolvida.

6.1 Dificuldades encontradas

No decorrer do projeto encontraram-se várias dificuldades nas quais se gastou mais dias do que o previsto. Das quais se destacam os problemas de comunicação entre o servidor Virtuoso e a aplicação, devido a sua pobre documentação para .NET e problemas de licenças para testes. Outra dificuldade foi a ambientação das bases de dados não relacionais para uma relacional.

6.2 Trabalho futuro

Como trabalho futuro seria importante compatibilizar todas as vistas de pesquisa numa só, visto que os colaboradores têm um grande interesse nessa vista.

Para além de melhorias, a empresa também ambicionava criar uma extensão do sistema para as ferramentas do Windows Ofice e arquivar os documentos logo na sua criação ou no caso do Outlook arquivar os e-mails ou ficheiros anexos caso o pretendam.

Por fim, seria importante integrar níveis de permissões de acesso e edição aos documentos por departamento. No momento não existem restrições, todos os utilizadores podem consultar, editar e criar documentos.

Referências

[1] Doug Miles. State of the ECM Industry 2011. página 29, 2011. URL: http://www.aiim.org/pdfdocuments/IW_ECM_State-of-Industry_2011.pdf.

[2] Corporation for National Research Initiatives., D-Lib Forum., e Federal High Performance Computing Program (U.S.). Information Infrastructure Technology and Applications Task Group. *D-Lib magazine.* Corp. for National Research Initiatives, 1995.

[3] Lia Fernandes. Sistemas de gestão documental e workflow no contexto da gestão da qualidade. (12), 2012. URL: https://repositorio-aberto.up.pt/bitstream/10216/68363/1/000154732.pdf.

[4] Cristina Ribeiro. As Ciências Documentais e a Construção da Web Semântica. URL: https://repositorio-aberto.up.pt/bitstream/10216/67326/2/65849.pdf.

[5] Catarina Isabel Pinto Bandeira Veloso. *Sistemas de Gestão Documental e Gestão da Qualidade : relevância e modelo de implementação*. Tese de doutoramento, 2011.

[6] Bruce E Bargmeyer. METADATA STANDARDS AND METADATA REGISTRIES: AN OVERVIEW. URL: https://www.bls.gov/ore/pdf/st000010.pdf.

[7] Terezinha Batista de Souza, Maria Elizabete Catarino, e Paulo Cesar dos Santos. Metadados: catalogando dados na Internet. *Transinformação - ISSN 2318-0889*, 9(2):93–105, 2012.

[8] Stuart Weibel, John Kunze, Carl Lagoze, e Misha Wolf. Dublin core metadata for resource discovery. *Internet Engineering Task Force RFC*, página 222, 1998. URL: http://www.hjp.at/doc/rfc/rfc2413.html, doi:10.1017/CBO9781107415324.004.

[9] Tony Lawson. A Conception of Ontology. *The Cambridge Social Ontology*, páginas 1–24, 2004.

[10] Ralph M. Stair e George Walter Reynolds. *Principles of information systems : a managerial approach*. Course Technology/Thomson Learning, 2001. URL: https://books.google.pt/books/about/Principles_of_Information_Systems.html?id=t1xGAAAAYAAJ&redir_esc=y.

[11] Dora Maria, D E Oliveira Sim, e E S Ribeiro Pereira. Engenharia de Ontologias para Redes Colaborativas. *Environment*, 2007. URL: https://repositorio-aberto.up.pt/bitstream/10216/12572/2/Textointegral.pdf.

[12] Robert Meersman. Ontology Engineering -The DOGMA Approach 1 Introduction and motivation. URL: http://www.jarrar.info/publications/%5BJM07%5D.v5.pdf.

[13] T Sheeba, Reshmy Krishnan, e M Justin Bernard. An Ontology in Project Management Knowledge Domain. *International Journal of Computer Applications*, 56(5):975–8887, 2012.

[14] Mauricio B. Almeida e Marcello P Bax. Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção. *Ciência da Informação*, 32(3):7–20, 2003. URL: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0100-19652003000300002&lng=pt&nrm=iso&tlng=pt, doi:10.1590/S0100-19652003000300002.

[15] Van Nguyen. Ontologies and Information Systems: A Literature Survey. Relatório técnico, 2011. URL: http://digext6.defence.gov.au/dspace/bitstream/1947/10144/1/DSTO-TN-1002PR.pdf.

[16] Peter Spyns, Robert Meersman, e Mustafa Jarrar. Data modelling versus ontology engineering. *ACM SIGMOD Record*, 31(4):12, 2002. URL: http://sce2.umkc.edu/csee/leeyu/class/CS690L/Reference/2.Meersman.pdfhttp://portal.acm.org/citation.cfm?doid=637411.637413, doi:10.1145/637411.637413.

[17] Dr.R. Sivakumar V. Maniraj. Ontology Languages – A Review.

[18] Fabien Gandon e Guus Schreiber. RDF 1.1 XML Syntax, 2016. URL: http://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/.

[19] Ramanathan Guha e Dan Brickley. RDF Schema 1.1, 2014. URL: http://www.w3.org/TR/2014/REC-rdf-schema-20140225/.

[20] Manu Sporny, Gregg Kellogg, e Markus Lanthaler. Json-Ld 1.0, 2013. URL: https://www.w3.org/TR/json-ld/.

[21] Richard Cyganiak, David Wood, e Markus Lanthaler. RDF 1.1 Concepts and Abstract Syntax, 2014. URL: http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/, doi:10.1007/s13398-014-0173-7.2.

[22] Carlos Buil Aranda, Olivier Corby, Souripriya Das, Lee Feigenbaum, Paul Gearon, Birte Glimm, Steve Harris, Sandro Hawke, Ivan Herman, Nicholas Humfrey, Nico Michaelis, Chimezie Ogbuji, Matthew Perry, Alexandre Passant, Axel Polleres, Eric Prud'hommeaux, Andy Seaborne, e Gregory Todd Williams. SPARQL 1.1 Overview, 2013. URL: https://www.w3.org/TR/sparql11-overview/.

[23] W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview. *OWL 2 Web Ontology Language*, (December):1–7, 2012. URL: http://www.w3.org/TR/owl2-overview/.

[24] Michael K Smith, Chris Welty, e Deborah L McGuinness. OWL Web Ontology Language Guide, 2004. URL: http://www.w3.org/TR/owl-guide/, doi:10.1145/1295289.1295290.

[25] Martin Hepp. Chapter 1 Ontologues: State of the Art, Business Potential,and Grand Challenges. *Ontology Management: Semantic Web, Semantic Web Services, and Business Application*, páginas 3–22, 2007. doi:doi: 10.1007/978-0-387-69900-4_1.

[26] Bhaskar Kapoor e Savita Sharma. A Comparative Study Ontology Building Tools for Semantic Web Applications. *International journal of Web & Semantic Technology (IJWesT)*, 1(3), 2010. doi:10.5121/ijwest.2010.1301.

[27] Oscar Corcho, Mariano Fernández-López, e Asunción Gómez-Pérez. Methodologies, tools and languages for building ontologies. Where is their meeting point? *Data and Knowledge Engineering*, 46(1):41–64, 2003. doi:10.1016/S0169-023X(02)00195-7.

[28] by Emhimed Alatrash e Dušan D Tošić. Using Web Tools for Constructing an Ontology of Different Natural Languages Under the supervision of. 2013.

[29] Protege. File:Protege-OWL.jpg - Protege Wiki. URL: http://protegewiki.stanford.edu/wiki/File:Protege-OWL.jpg#file.

[30] V J Kawano. Desenvolvimento de uma Ontologia para Gerenciamento de Projetos. *monografiascicunbbr*, 2009. URL: http://monografias.cic.unb.br/dspace/bitstream/123456789/194/1/MonografiaProj2.pdf.

[31] David C Faye, Olivier Cure, e Guillaume Blin. A survey of RDF storage approaches. *ARIMA Journal ARIMA Journal* –, 15(12):11–35, 2012.

[32] OpenLink. OpenLink Virtuoso Home Page. URL: https://virtuoso.openlinksw.com/.

[33] About | DBpedia. URL: http://wiki.dbpedia.org/about.

[34] Stardog. Stardog 5: The Manual. URL: http://www.stardog.com/docs/.